REMARKS

In response to the Office Action dated September 26, 2003, Applicant respectfully requests reconsideration. To further the prosecution of the application, each of the rejections set forth in the Office Action has been considered and is addressed below.

Claims 1-9 are pending in the application. Claim 8 has been amended solely for clarification and not to distinguish over the prior art of record.

Specification Objections

The Abstract of the Disclosure is objected to for exceeding 150 words. Applicant presents a replacement Abstract herein which addresses this objection. Accordingly, Applicant respectfully requests that the objection to the Abstract be withdrawn.

Rejections Under 35 U.S.C. §102

Claims 1-9 stand rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent No. 6,182,283 to Thomson (herein referred to as "Thomson"). Applicant respectfully traverses this rejection.

1. Brief Overview Of Thomson

Thomson is directed to removing uncallable virtual functions from a compiled and linked program (col. 2, lines 63-65). Thomson is concerned with the linking of object modules compiled from an object-oriented program having "virtual functions". Thomson removes references to unused virtual functions from the executable code (col. 3, lines 6-7). This is achieved by the linker using information tagged in the modules by the compiler about their use of virtual functions to determine whether the program calls a virtual function (col. 3, lines 7-11 and col. 5, lines 5-7). If there can be no call invoking the virtual function, the linker excludes the virtual function from the resulting executable program (col. 5, lines 9-11).

In Thomson, calls to virtual functions are implemented as indirect calls through a virtual function table ("VFT") (col. 1, lines 53-55). Each object whose class contains virtual functions has a pointer to one or more VFTs. If a class introduces a virtual function, the compiler allocates a virtual function table for the class (col. 1, lines 59-61). An object has a data structure that

contains the layout of its data members and a pointer to a VFT which is initialized to point to the VFT associated with the class (col. 1, line 65-col. 2, line 2). In Figure 9, for example, object "a" points to the VFT of class "A" which in turn references virtual functions that should be called for that object (Figure 9 and col. 5, lines 61-62). Thomson aims to eliminate unnecessary linking for virtual functions that will never be called (col. 7, lines 22-31).

Figure 10 of Thomson shows object modules 1-5, each containing a different virtual function and modules 1, 3, and 4 also containing a VFT for referencing another virtual function contained in a different object code (Figure 10). Figure 11a depicts a conventional system which shows object modules 1-5 as linked, including object modules 2 and 5 despite such modules not being called during execution. Figure 11b shows that the system of Thomson reduces the size of the linked executable program by detecting virtual functions that cannot be called from the application and not linking those modules which contain the detected virtual functions (i.e., modules 2 and 5).

2. Amended Claim 8 Patentably Distinguishes Over Thomson

Amended claim 8 recites a method of assembling an object code module for linking to form an executable program, the method comprising: executing a set of assembler directives including a macro call directive, and responsive to the macro call directive: naming a location in a macro section in the object code module containing a plurality of code sequences, at least some of which are likely to be repeatedly included in the executable program; marking at an insertion location IL in a set of section data in the object code module where at least some of said code sequences are to be inserted in the final executable program; generating in association with the section data a macro call relocation identifying the named location in the macro section; and generating a set of macro relocations associated with said macro section for selecting said at least some code sequences for insertion at the insertion location.

The Office Action cites a passage (col. 4, lines 31-32) wherein Thomson purportedly discloses executing a set of *assembler directives* including a macro call directive, as recited in claim 8. Thomson, unlike claim 8, is not concerned with the compilation or assembly of object code modules, but rather is concerned with the *linking* of object modules compiled from an object-oriented program having "virtual functions". Although the passage relied on by the

Office Action states that a program is compiled and linked into an executable file, Thomson merely describes the standard way of compiling source code wherein the object code modules consist of section data (i.e., machine code and data generated by the compiler) and control information including descriptions of relocation actions (col. 4, lines 32-42). There is no disclosure in Thomson of generating an object code module comprising a separate macro section having code sequences and associated macro relocations, as claimed. Moreover, Thomson does not disclose or suggest, in the cited passage or elsewhere, *executing a set of assembler directives including a macro call directive*, as recited in claim 8.

The Office Action asserts that Figure 10 of Thomson discloses naming a location in a macro section in the object code module containing a plurality of code sequences, at least some of which are likely to be repeatedly included in the executable program, as claimed. Figure 10 shows object modules 1-5, each containing a different virtual function, and modules 1, 3, and 4 also containing a VFT for referencing another virtual function contained in a different object module. However, Figure 10 does not disclose or suggest naming a location in a *macro section* in the object code module containing a plurality of code sequences, at least some of which are likely to be repeatedly included in the executable program, as recited in claim 8.

The Office Action asserts that Figure 11b shows *marking at an insertion location IL in a set of section data* in the object code module where at least some of said code sequences are to be inserted in the final executable program, as claimed. Figure 11b shows that the system of Thomson operates by detecting virtual functions that cannot be called from the application and by not linking those modules (e.g., modules 2 and 5 in Figure 10) which contain the detected virtual functions. However, Figure 11b of Thomson does not disclose or suggest a macro section having sets of code sequences that can be *inserted at an identified insertion location* in said section data of the object code module or, for that matter, the linked main program. Hence, Thomson does not disclose or suggest, in the cited figure or elsewhere, *marking at an insertion location IL in a set of section data* in the object code module where at least some of said code sequences are to be inserted in the final executable program, as recited in section 8.

The Office Action cites a passage (col. 6, lines 7-18) wherein Thomson purportedly discloses generating in association with the section data a macro call relocation identifying the named location in the macro section, as claimed. The cited passage describes Figure 9 which

shows the relationship between objects and the VFTs constructed by the compiler. In particular, Figure 9 shows object "a" pointing to the VFT of class "A" which in turn references virtual functions that should be called for that object (Figure 9, col. 5, lines 61-62 and col. 6, lines 7-12). Thus, the pointer references in Thomson do not relate to generating in association with the section data *a macro call relocation identifying the named location in the macro section*, as recited in claim 8. The pointers are simply pointers to virtual functions.

The Office Action asserts that Figure 5 of Thomson shows generating a set of relocations for selecting said at least some code sequences for insertion at the insertion location. In Figure 5, Thomson shows a process for finding relocations that apply to the current code or data record and setting a field in the linker symbol table indicating whether the virtual function is referenced or not (col. 14, line 62 to col. 15, line 20). Applicant has amended claim 8 to recite generating a set of *macro relocations associated with said macro section* for selecting said at least some code sequences for insertion at the insertion location. The teaching in Thomson of finding relocations and flagging whether a virtual function is referenced or not does not teach or suggest generating a set of *macro relocations associated with said macro section* for selecting said at least some code sequences for insertion at the insertion location, as claimed.

Amended claim 8 identifies two different sets of relocations corresponding to the normal section data part of the object code module and the macro section. Specifically, in addition to the normal relocations, amended claim 8 recites "a set of macro relocations associated with said macro section". Thus, claim 8 further distinguishes over Thomson which does not disclose two different sets of relocations, but rather, Thomson discloses only one set of relocations.

For each of the reasons discussed above, Thomson does not anticipate claim 8. Accordingly, Applicant respectfully requests withdrawal of the rejection of claim 8 under 35 U.S.C. §102(e).

3. Claims 1-7, and 9 Patentably Distinguish Over Thomson

The Office Action rejects independent claims 1, 5, and 9 for the same reasons as claim 8. Applicant notes that claims 1, 5, and 9 pertain to subject matter different from claim 8 despite containing some common features of claim 8. In particular, independent claim 8 is directed to a method of *assembling* an object code module whereas independent claims 1, 5, and 9 relate to a

linking operation. Specifically, Figure 6 of the present application shows an assembler 2 which is responsible for assembling an object code module 3 and a linker 4 which is responsible for linking the various object code modules 3 to form an executable program 5. Independent claim 8 is directed to the functionality of assembler 2, while independent claims 1, 5, and 9 are directed to the linker 4. Hence, the rejection of independent claims 1, 5, and 9 on the same basis as claim 8 is inappropriate.

3.1 Claim 1 Patentably Distinguishes Over Thomson

Claim 1 is directed to a method of preparing an executable program from a plurality of object code modules, each module containing sets of section data and associated relocations, and at least one of said modules further including *a macro section containing code sequences* at least some of which are likely to be repeatedly included in the executable program *and macro relocations associated with said macro section*, wherein at least one of said sets of section data includes at least one *insertion location where said code sequences are to be inserted* and its associated relocation instructions include a macro call relocation (R_CALL_MACRO) identifying a location in the macro section, the method comprising at link time when said executable program is prepared: reading said sets of section data and relocation instructions; on locating said macro call relocation identifying the location in the macro section; and *inserting said at least some code sequences from said location in the macro section into the set of section data at the insertion location*, said at least some code sequences being selected by reading the macro relocations (emphasis added).

Thomson does not teach or suggest a macro section containing code sequences and macro relocations, and section data including an insertion location where code sequences are to be inserted, as required by claim 1.

Moreover, claim 1 identifies two different sets of relocations corresponding to the normal section data part of the object code module and the macro section. Specifically, in addition to the normal relocations corresponding to sets of section data, claim 1 recites "macro relocations associated with said macro section". Thus, claim 1 further distinguishes over Thomson which does not disclose two different sets of relocations, but rather, Thomson discloses only one set of relocations.

Furthermore, Thomson does not disclose or even remotely suggest *identifying the location in the macro section and inserting at least some code sequences from said location in the macro section into the set of section data at the insertion location, said at least some code sequences being selected by reading the macro relocations*, as required by claim 1.

For these reasons and the reasons discussed above in connection with claim 8, the rejection of claim 1 under 35 U.S.C. §102(e) should be withdrawn.

Claims 2-4 depend from claim 1 and are allowable for at least the same reasons.

3.2 Claim 5 Patentably Distinguishes Over Thomson

Claim 5 is directed to a linker for preparing an executable program for a plurality of object code modules, each module containing sets of section data and relocations, at least one of said modules further including *a macro section containing code sequences* at least some of which are likely to be repeatedly included in the executable program *and macro relocations associated with said macro section*, wherein at least one of said sets of section data includes at least one *insertion location where said code sequences are to be inserted*, and its associated relocations include a macro call relocation identifying a location in the macro section, the linker comprising: a relocation module for reading the relocations, the relocation module being operable to identify a macro call relocation and to locate said location in the macro section; *a section data module for holding section data to which the relocations relate and arranged to receive said at least some code sequences from the location in the macro section to be inserted at the insertion location*; and a program preparing means which prepares said executable program including said set of section data with the inserted code sequences (emphasis added).

Thomson does not teach or suggest a macro section containing code sequences and macro relocations, and section data including an insertion location where code sequences are to be inserted, as required by claim 5.

Moreover, claim 5 identifies two different sets of relocations corresponding to the normal section data part of the object code module and the macro section. Specifically, in addition to the normal relocations corresponding to sets of section data, claim 5 recites "sets of macro relocations associated with said macro section". Thus, claim 5 further distinguishes over

Thomson which does not disclose two different sets of relocations, but rather, Thomson discloses only one set of relocations.

Furthermore, Thomson does not disclose or suggest a relocation module that is operable to identify a macro call relocation and to locate the location in the macro section and a section data module that is arranged to receive at least some code sequences from the location in the macro section to be inserted at the insertion location, as required by claim 5.

For these reasons and the reasons discussed above in connection with claims 1 and 8, the rejection of claim 5 under 35 U.S.C. §102(e) should be withdrawn.

Claims 6-7 depend from claim 5 and are allowable for at least the same reasons.

3.3 Claim 9 Patentably Distinguishes Over Thomson

Claim 9 is directed to a computer program product in the form of an object code module which contains sets of section data and relocations, the module further including *a macro section containing code sequences* at least some of which are likely to be repeatedly included in the executable program *and a set of macro relocations associated with said macro section* wherein at least one of said sets of section data includes an *insertion location where said code sequences are to be inserted*, and its associated relocations include a macro call relocation identifying a location in the macro section, the computer program product being cooperable with a linker to cause execution of relocation operations by the linker in dependence on said relocations and including *identifying the location in the macro section and inserting said at least some code sequences from that location in the macro section in the set of section data at the insertion location* (emphasis added).

Thomson does not teach or suggest a macro section containing code sequences and macro relocations, and section data including an insertion location where code sequences are to be inserted, as required by claim 9.

Moreover, claim 9 identifies two different sets of relocations corresponding to the normal section data part of the object code module and the macro section. Specifically, in addition to the normal relocations corresponding to sets of section data, claim 9 recites "sets of macro relocations associated with said macro section". Thus, claim 9 further distinguishes over

Thomson which does not disclose two different sets of relocations, but rather, Thomson discloses only one set of relocations.

For these reasons and the reasons discussed above in connection with claims 1, 5, and 8, the rejection of claim 9 under 35 U.S.C. §102(e) should be withdrawn.

CONCLUSION

In view of the foregoing amendments and remarks, this application should now be in condition for allowance. A notice to this effect is respectfully requested. If the Examiner believes, after this amendment, that the application is not in condition for allowance, the Examiner is requested to call the Applicant's attorney at the telephone number listed below to discuss any outstanding issues relating to the allowability of the application.

If this response is not considered timely filed and if a request for an extension of time is otherwise absent, Applicant hereby requests any necessary extension of time. If there is a fee occasioned by this response, including an extension fee, that is not covered by an enclosed check, please charge any deficiency to Deposit Account No. 23/2825.

Respectfully submitted,
Richard Shann, Applicant

By: William R. McClellan

William R. McClellan, Reg. No.: 29,409
Wolf, Greenfield & Sacks, P.C.
600 Atlantic Avenue
Boston, Massachusetts 02210-2211
Telephone: (617) 720-3500

Docket No. S1022.80522US00
Date: December 29, 2003
x12/26/03x